



# An Innovative High-Performance Architecture for Vector and Matrix Math Algorithms

***Presented by: Tim Olson, Architect***

**HPEC 2002 – September 24, 2002**

***Authors: Veeraraghavan Anantha, Ph.D.;  
Christophe Harlé, Ph.D.; Tim Olson, George Yost, Ph.D.***

© 2002 Intrinsity, Inc.

Intrinsity, the Intrinsity logo, the Intrinsity dot logo, Advanced Signal Processor, and FastMATH are trademarks of Intrinsity, Inc. MIPS is among the registered trademarks and MIPS32 is among the trademarks of MIPS Technology, Inc. RapidIO and the RapidIO logo are trademarks of the RapidIO Trade Association. All other trademarks are for reference purposes only and are the property of their respective owners.

| Report Documentation Page  |                                    |                                     |   | Form Approved<br>OMB No. 0704-0188       |                                 |
|--|------------------------------------|-------------------------------------|---|--|---------------------------------|
| Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. |                                    |                                     |   |  |                                 |
| 1. REPORT DATE<br><b>24 SEP 2002</b>   |                                    | 2. REPORT TYPE<br><b>N/A</b>        |   | 3. DATES COVERED<br><b>-</b>             |                                 |
| 4. TITLE AND SUBTITLE<br><b>An Inovative High-Performance Architecture for Vector and Matrix Math Algorithms</b>   |                                    |                                     |   | 5a. CONTRACT NUMBER                      |                                 |
|  |                                    |                                     |   | 5b. GRANT NUMBER                         |                                 |
|  |                                    |                                     |   | 5c. PROGRAM ELEMENT NUMBER               |                                 |
| 6. AUTHOR(S)   |                                    |                                     |   | 5d. PROJECT NUMBER                       |                                 |
|  |                                    |                                     |   | 5e. TASK NUMBER                          |                                 |
|  |                                    |                                     |   | 5f. WORK UNIT NUMBER                     |                                 |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br><b>Intrinsity, Inc.</b>  |                                    |                                     |   | 8. PERFORMING ORGANIZATION REPORT NUMBER |                                 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  |                                    |                                     |   | 10. SPONSOR/MONITOR'S ACRONYM(S)         |                                 |
|  |                                    |                                     |   | 11. SPONSOR/MONITOR'S REPORT NUMBER(S)   |                                 |
| 12. DISTRIBUTION/AVAILABILITY STATEMENT<br><b>Approved for public release, distribution unlimited</b>  |                                    |                                     |   |  |                                 |
| 13. SUPPLEMENTARY NOTES<br><b>Also see ADM001473 , The original document contains color images.</b>  |                                    |                                     |   |  |                                 |
| 14. ABSTRACT   |                                    |                                     |   |  |                                 |
| 15. SUBJECT TERMS  |                                    |                                     |   |  |                                 |
| 16. SECURITY CLASSIFICATION OF:  |                                    |                                     | 17. LIMITATION OF ABSTRACT<br><b>UU</b> | 18. NUMBER OF PAGES<br><b>12</b>         | 19a. NAME OF RESPONSIBLE PERSON |
| a. REPORT<br><b>unclassified</b>   | b. ABSTRACT<br><b>unclassified</b> | c. THIS PAGE<br><b>unclassified</b> |   |  |                                 |

# Intrinsity FastMATH™ Vector and Matrix Math Processor

**Optimized for real-time and adaptive signal processing needs:**

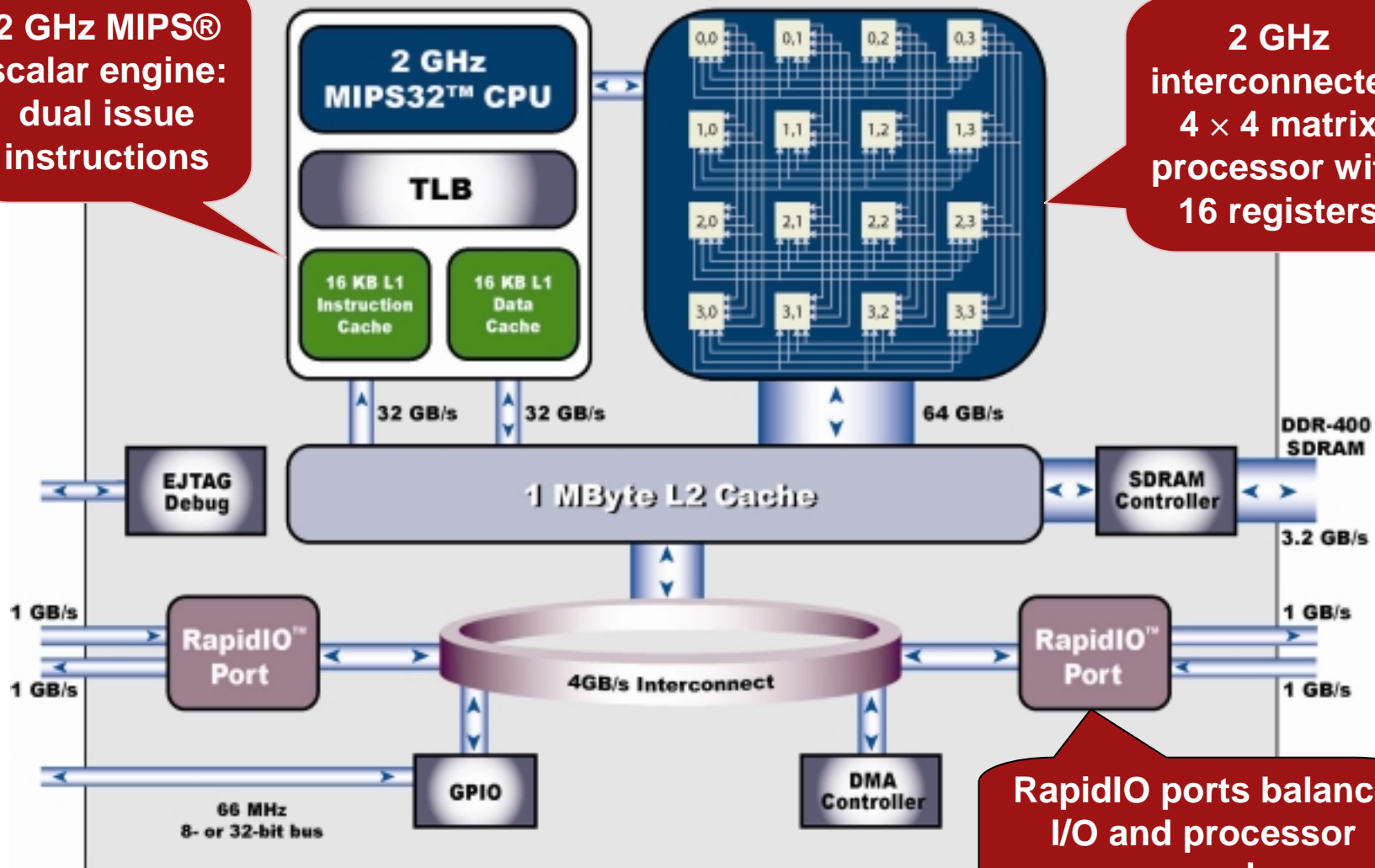
**Innovative architecture:**

- *2 GHz SIMD  $4 \times 4$  matrix engine with multiprocessor scalability due to high bandwidth RapidIO™ interfaces*
- **Fixed-point math**
- **High-level (e.g., C) language programmable**
  - **Compiler built-in matrix intrinsics**
  - **Vector/matrix library**
- **On-chip matrix coprocessor and MIPS32™ ISA RISC core**
- **$4 \times 4$  array of processors, each with sixteen 32-bit registers, two 40-bit MACs**
- **64 GOPS (peak)**
- **Matrix and vector math native instructions: 1-, 8-, 16-, 32-bit support; convenient complex math**
- **Descriptor-based DMA controller**
- **1 Mbyte on-chip cache-coherent L2 cache**

**Speed *plus* an architecture designed for parallel computations**

# Intrinsity FastMATH Vector and Matrix Math Processor

**2 GHz MIPS® scalar engine: dual issue instructions**

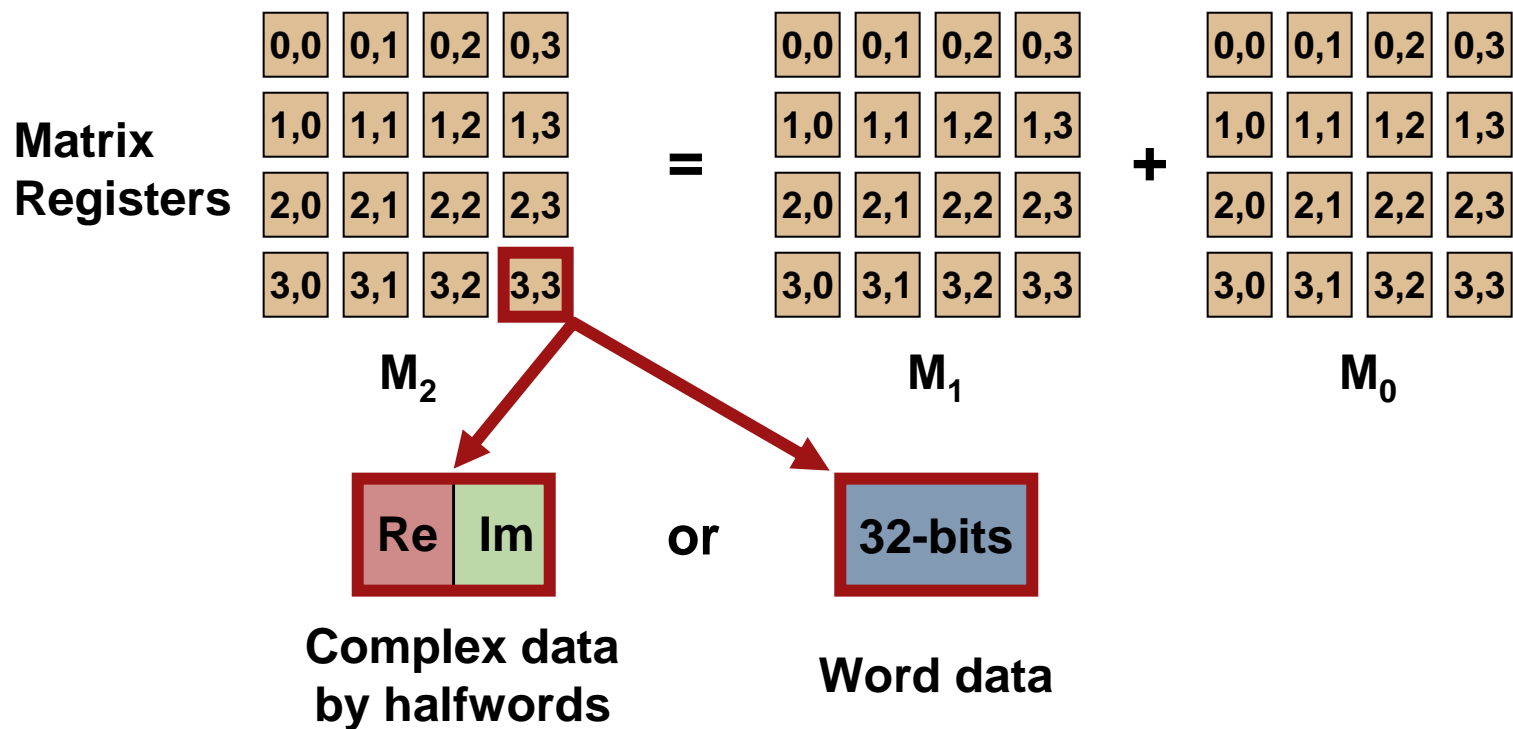


**RapidIO ports balance I/O and processor speed**

# Matrix Register Arithmetic: Element-by-Element

The matrix engine has 16 matrix registers, each with 16 32-bit values.  
Halfword and word arithmetic is supported.

Single instruction, element-wise addition of two  $4 \times 4$  matrices



# Matrix Register Arithmetic: Matrix Multiplication

**Matrix-multiply of two  $4 \times 4$  submatrices by halfword, for example to support 16-bit complex arithmetic**

**One instruction**

- **Four cycles**  
(2 ns @ 2 GHz)
- **128 operations**

$$\sum_{k=0}^3 M0^h(0,k) \times M1^h(k,0)$$

**matmulhh.m.m M2,M0,M1**

```
for i = 0 to 3
  for j = 0 to 3
    sum = 0
    for k = 0 to 3
      sum = sum + M0h(i,k) × M1h(k,j);
    M2h(i,j) = sum;
```

**High-high halfword multiply, e.g., re × re**

**Matrix Registers**

|     |     |     |     |
|-----|-----|-----|-----|
| 0,0 | 0,1 | 0,2 | 0,3 |
| 1,0 | 1,1 | 1,2 | 1,3 |
| 2,0 | 2,1 | 2,2 | 2,3 |
| 3,0 | 3,1 | 3,2 | 3,3 |

**M<sub>2</sub>**

=

|     |     |     |     |
|-----|-----|-----|-----|
| 0,0 | 0,1 | 0,2 | 0,3 |
| 1,0 | 1,1 | 1,2 | 1,3 |
| 2,0 | 2,1 | 2,2 | 2,3 |
| 3,0 | 3,1 | 3,2 | 3,3 |

**M<sub>0</sub>**

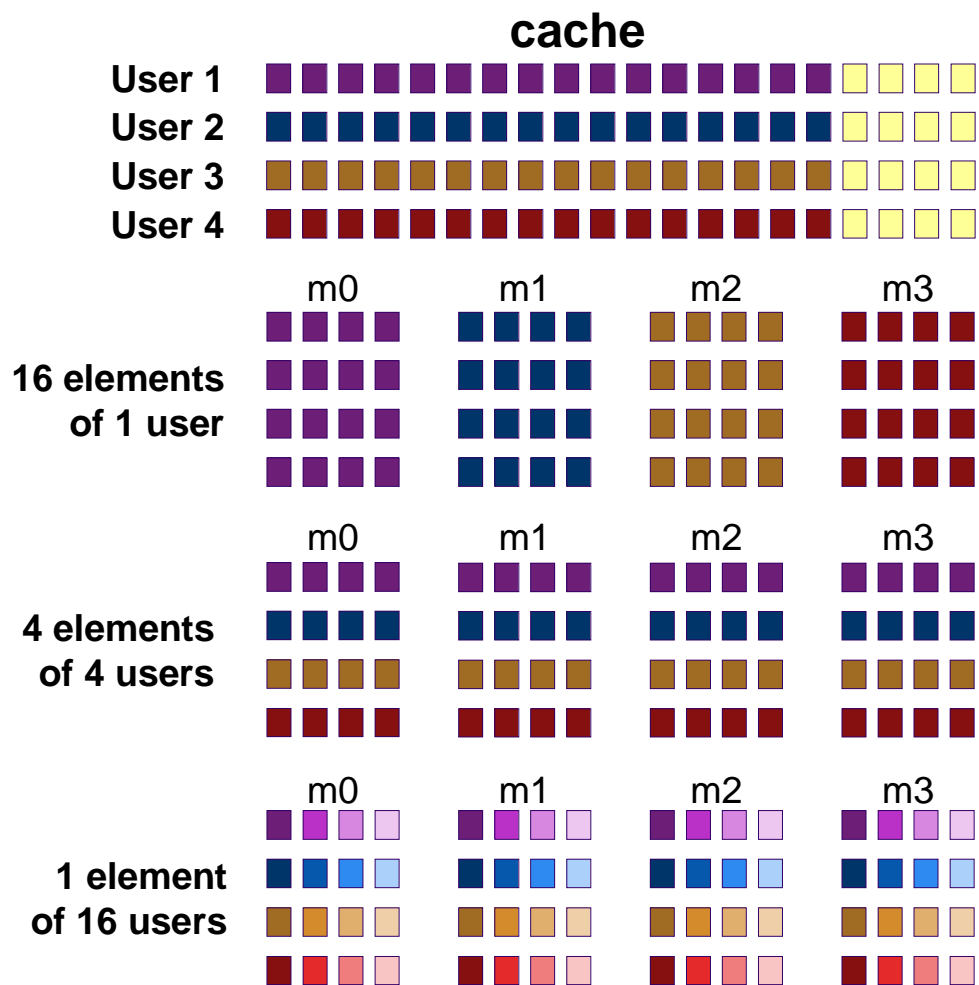
×

|     |     |     |     |
|-----|-----|-----|-----|
| 0,0 | 0,1 | 0,2 | 0,3 |
| 1,0 | 1,1 | 1,2 | 1,3 |
| 2,0 | 2,1 | 2,2 | 2,3 |
| 3,0 | 3,1 | 3,2 | 3,3 |

**M<sub>1</sub>**

**Can subdivide large matrices into  $4 \times 4$  parts for multiplication**

# Matrix Register Arithmetic: Block Rearrangement for Parallelism



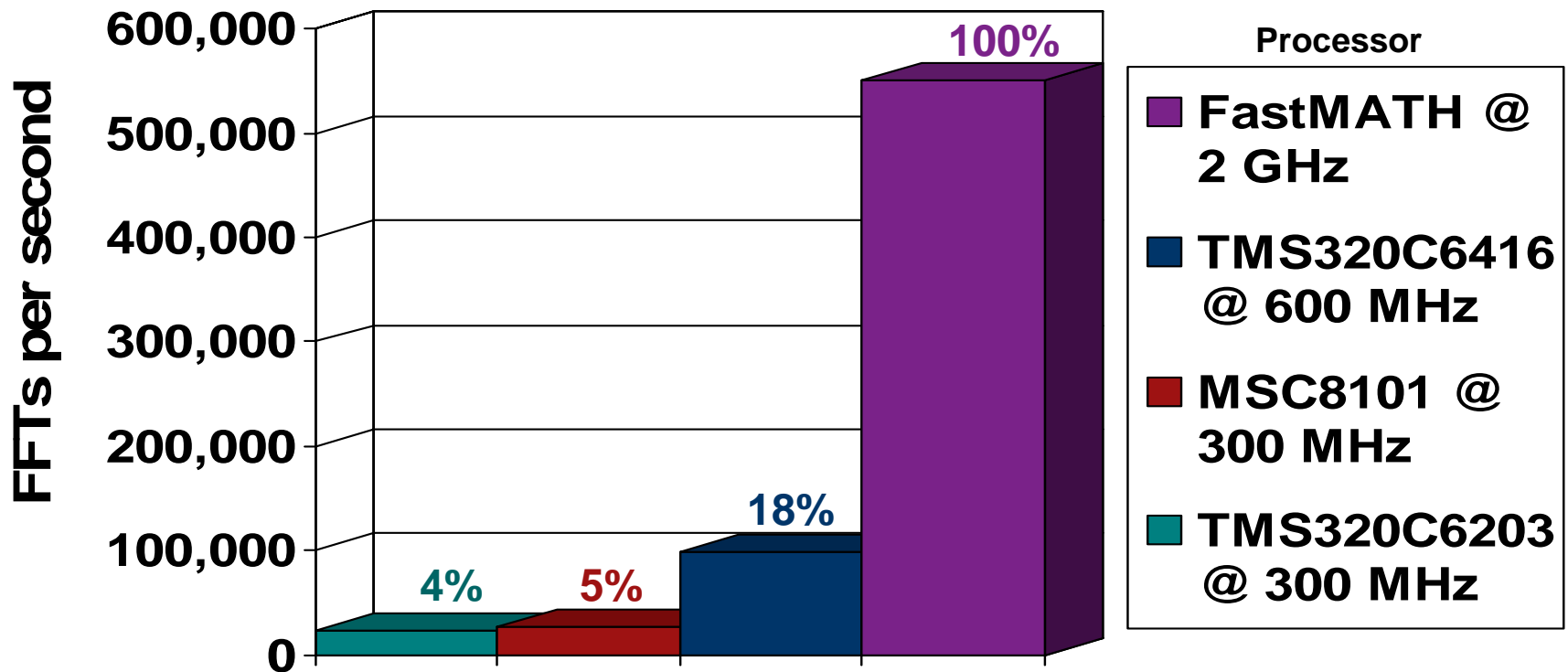
**Load 4 or 16 data streams  
(users) and re-block for  
SIMD parallel processing**

- Original register load instructions
- **block4** (four cycles):  
matrix operations on four streams
- For SIMD operations on **16 parallel data streams**:  
continue rearrangement with block data movement instructions—70 cycles (35 ns) total

## FastMATH Performance Example: Fast Fourier Transform

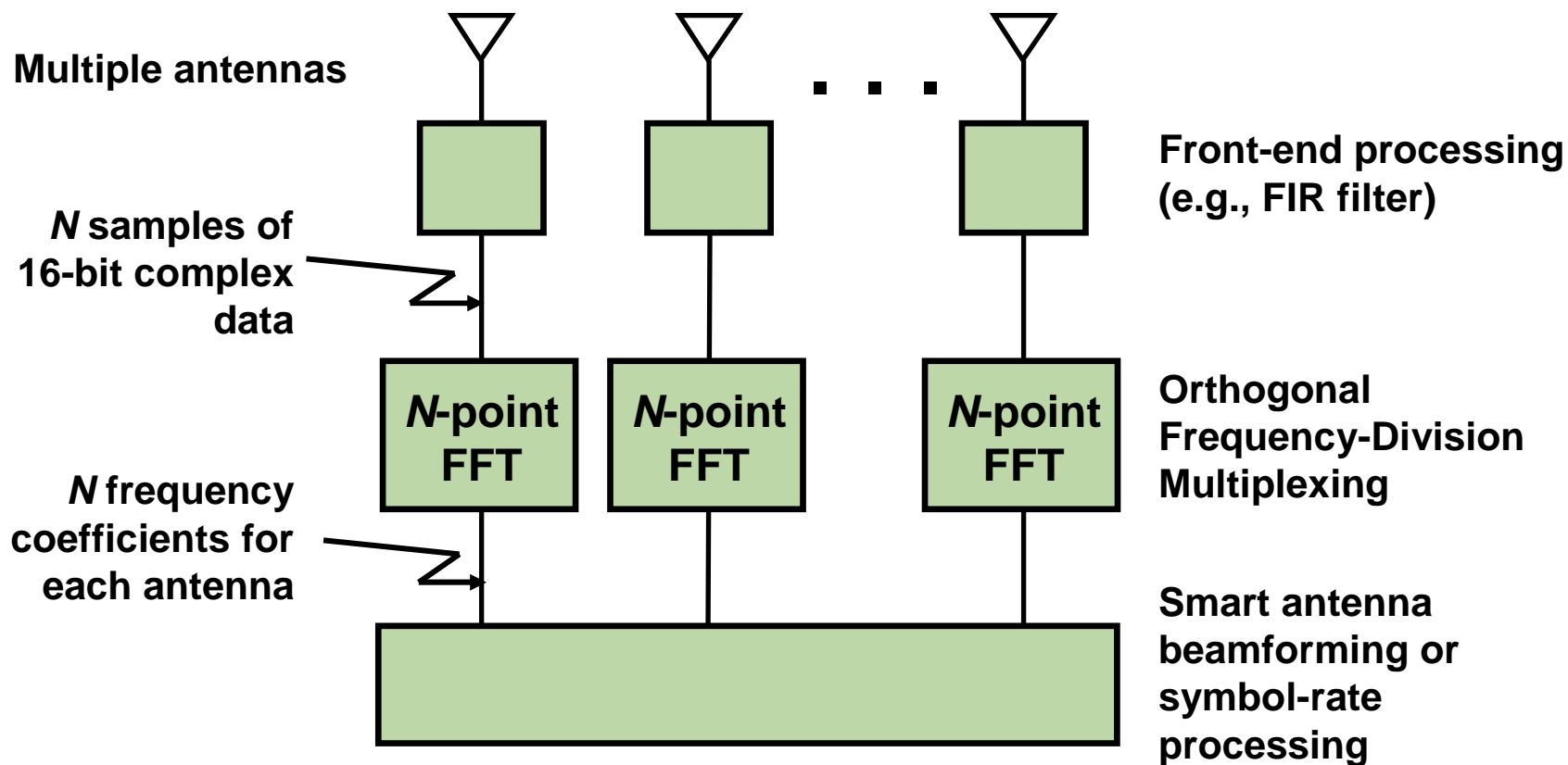
**Matrix** architecture plus cycle speed combine approximately equally for advantage on this key benchmark

**1 K Radix-4 FFT, 16-bit complex data**



*Notes: Competitive data from published benchmarks  
Competitive clock rates are highest announced*

# FastMATH Performance Example: FFT to Implement OFDM

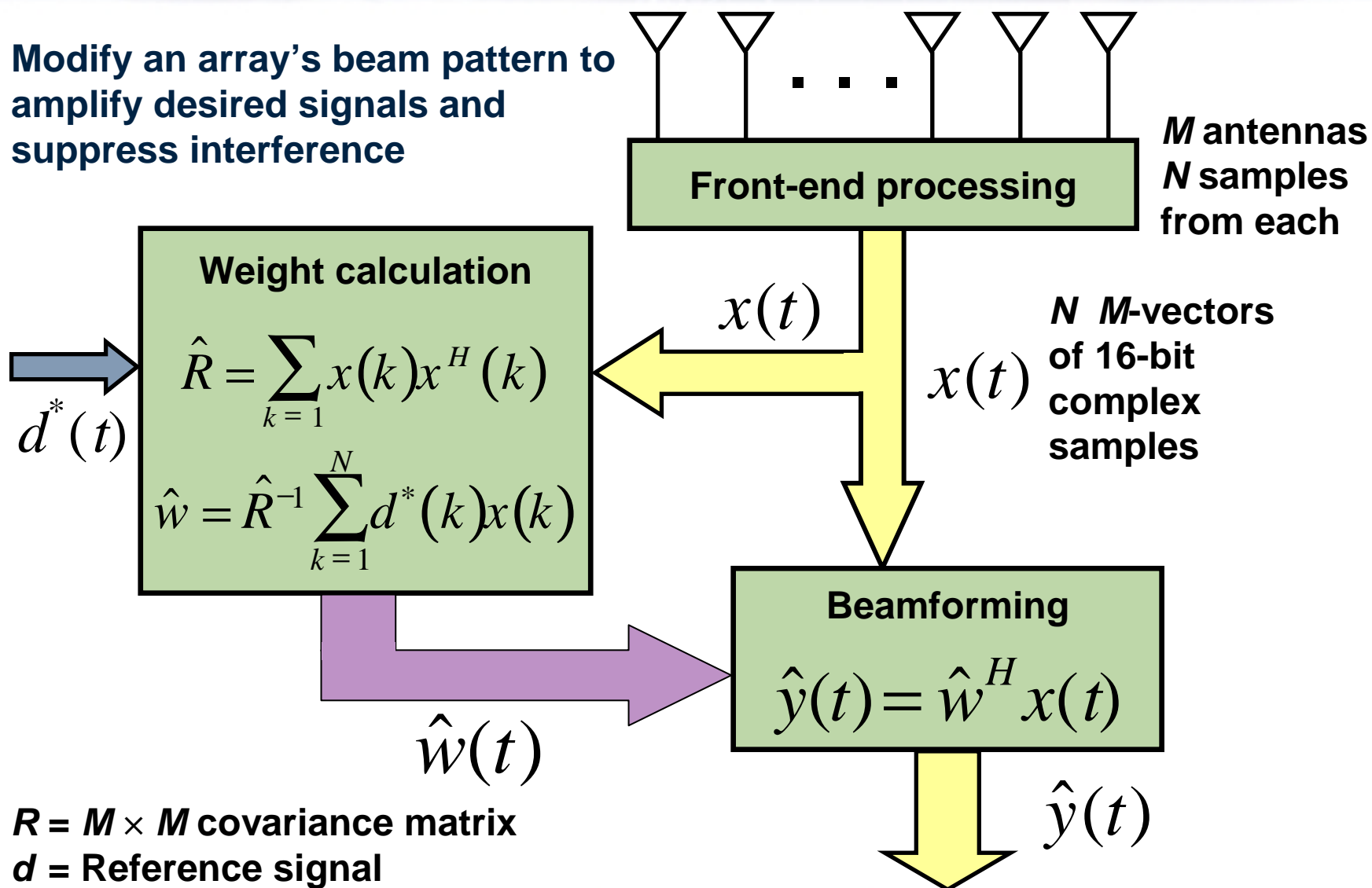


## Example results:

for 8 antennas, 10 Msamples per second, 1024-pt complex FFT:  
requires **14.4% FastMATH processor**

# FastMATH Performance Example: Smart Antennas

Modify an array's beam pattern to  
amplify desired signals and  
suppress interference



# FastMATH Performance Example: Smart Antennas

## Background

- More users than antennas  $\Rightarrow$  orthogonal beams not possible
- No a priori information about signal directions  $\Rightarrow$  need real-time adaptation
- Input stream is 16-bit complex data

## FastMATH Implementation

- Covariance matrix calculated by *complex matrix-matrix multiplications on  $4 \times 4$  submatrices*, then re-assembling full matrix
- Covariance matrix inverted by Cholesky decomposition; use *block matrix manipulation* instructions to rearrange input into blocks for SIMD parallelization
- Beamforming using matrix-matrix multiplications; more efficient than simple vector math

## WCDMA Example Results

- With 64 voice users and 16 antennas, 4 rake fingers per user, weights updated every slot: 0.73 FastMATH processors

## Scaled Multiprocessor Example: CDMA Multi-User Detection

### Algorithms

- *Mitigate interference between users in CDMA*
- Solve for estimators for correct symbols, beginning with user-user correlation matrix  $R$  and user input vector  $y$
- Difference equation for interference on symbol  $m$  of desired user from near-by symbols of all other users:

$$y_m = \sum_{k=-K}^K R_{m-k} \hat{b}_{m-k}$$

- $\hat{b}$  is desired estimator vector for symbol  $m$  of  $N$  users to be found

### Implementation

- *Jacobi iteration*: Solve for matrix  $B$  of  $M$  symbols for  $N$  users. Perform matrix-matrix multiplications distributed over processors
- Calculate correlation matrices  $R$  on chip; large capacity L2 cache reduces data transfer
- At each iteration exchange partial results over RapidIO port via DMA
- RapidIO interfaces work in background in parallel with computations – data transfer time efficiently hidden

## Scaled Multiprocessor Example: WCDMA Short Code Multi-User Detection

- Data transfer in parallel with computation
- Scalable multiprocessor system distributing tasks and results over RapidIO interface via coherent L2 cache

